EFFICIENT INTEGRATION OF OLD AND NEW RESEARCH TOOLS FOR AUTOMATING THE IDENTIFICATION AND ANALYSIS OF SEISMIC REFERENCE EVENTS

Wilmer Rivers,[1] Robert A. Wagner,[1] Eric Siu,[1] Joey Chen,[1] Craig A. Schultz,[2] Douglas A. Dodge,[2] and Gregory Pope[2]

Multimax, Inc.;[1] Lawrence Livermore National Laboratory[2]

Sponsored by National Nuclear Security Administration
Office of Nonproliferation Research and Engineering
Office of Defense Nuclear Nonproliferation

Contract No. DE-FG02-01ER83218[1] and W-7405-ENG-48[2]

ABSTRACT

The selection and study of reference events for inclusion in the National Nuclear Security Administration (NNSA) Knowledge Base requires the application of a much broader suite of seismic analysis software than does either the routine production of a seismic bulletin or the subsequent preliminary screening of those bulletin events to conduct nuclear monitoring. For either of those latter applications, a large program designed explicitly for that single purpose can be applied to the many events that will be processed daily, but to study the much smaller number of reference events in adequate detail it is necessary to use many separate specialized programs. These individual programs must communicate with one another in an efficient and flexible manner, so that a scientist can select which programs to run and in what order, thereby improvising a data-processing pipeline of programs passing their results from one to the next. The scientist may in fact wish to develop new software to perform specific analyses on the particular events under examination, and this new software must work in tandem with the existing software.

We have therefore explored the use of a distributed programming environment as the architecture for a reference event analysis system. This environment was chosen so that specialized analysis tools could be developed independently of one another and then integrated with existing programs. The specialized tools would remain separate from the existing software, however, and they could in fact run on different platforms, communicating with the other software across a local network or even the Internet. A distributed processing environment allows an expanded system to be built from stand-alone software components that can easily be added to or removed from the existing system by re-directing the data communications without requiring fundamental changes to the existing software. A particular advantage to this approach is that a researcher who wishes to design and test a particular algorithm for seismic analysis can construct a software component that performs only that algorithm, and then existing software systems can be used to provide data input and graphics display capabilities for testing that component. It will thus not be necessary to include those capabilities within the component under development nor to make changes to the existing software systems that could introduce harmful software "side effects" to them.

In the first phase of this project, we investigated implementing this system architecture using the large seismic analysis package *geotool* as a client program that invoked services from distributed components via the Common Object Request Broker Architecture (CORBA) as the data communications platform and the Internet Inter-ORB Protocol (IIOP) as the wire protocol for Internet transmission. CORBA is software architecture for binary data communications, and it was designed in the 1990s primarily for client-server computing within a single enterprise. Because the data transmissions are binary, CORBA cannot be used to invoke services across a firewall. Within the last two years there has emerged a new software standard for distributed computing, especially among different enterprises communicating across the Internet, that is based on ASCII messages (encoded using XML) rather than binary transmissions. This standard, known as Simple Object Access Protocol (SOAP), allows components to operate as "Web services" that can be called using HTTP as the wire protocol and that can thus operate across firewalls. In our recent work we have replaced CORBA with SOAP as the basis for our distributed data processing. Because Web services are becoming an important aspect of electronic commerce, most modern software development systems on the UNIX, Java, and Windows platforms all include tools that facilitate the construction of Web services, and since the data are transmitted as ASCII messages rather than binary code, software components on these different platforms can operate together as a single data processing system. This inter-platform distribution of components is likely to become a key aspect of most newly developed large data analysis systems.

OBJECTIVE

Scientists use a variety of stand-alone computer programs to analyze and identify seismic events for nuclear explosion monitoring, and an especially wide selection of software tools is needed for the detection and intensive study of reference events that will be included in the NNSA Knowledge Base for use in future event comparisons. Some of these stand-alone programs are large software packages that offer many tools for routine seismic analysis, but they are difficult to modify to include additional tools for specialized tasks. Others of these stand-alone programs offer the capability of performing only specific operations, and they must be used in conjunction with other programs such as interactive waveform graphics displays that offer more general analysis capabilities. In most cases neither the large software packages nor the specialized analysis programs can communicate adequately from one to another without the tedious creation and input of temporary data files and other awkward techniques. Since the stand-alone programs cannot exchange data easily, it is difficult to use them in a data-processing pipeline that could add new capabilities to those offered by the large software packages or that could allow the specialized analysis programs to rely on other software for tasks such as graphics displays.

A reference event analysis system should therefore be built by using a system architecture that facilitates data flow among these stand-alone programs, including any new programs that will be developed in the course of future research and that may be especially valuable for the identification and characterization of reference events. This new architecture should allow the results of one program to be sent easily to another one, as chosen on a case-by-case basis by the seismic analyst, without the creation of temporary files and database tables. Because many of the stand-alone seismic analysis programs that need to communicate with one another are written in different computer languages, and many are written for use under different operating systems, it will be important for this architecture to be as nearly platform-independent as possible. Furthermore, the communications among the separate programs should allow access to remote resources for data retrieval or specialized computations. The reference event analysis system should therefore be constructed as a distributed system of individual software components rather than as a single large software package. The first objective of our study is to examine software architectures and communications protocols that will allow existing and newly developed programs to be used as the components in this distributed system. Our next objective is to select and modify those programs so that they can be integrated into a reference event analysis system using that distributed system architecture.

RESEARCH ACCOMPLISHED

It is our intention to use the familiar seismic analysis program *geotool* as the cornerstone, at least initially, of the reference event system because it offers a wide suite of graphics tools and processing routines. The *geotool* routines will first be supplemented and eventually be replaced by other processing routines that will operate remotely, and finally the core graphics will also be replaced by a dedicated user interface program. We have therefore undertaken tasks to augment *geotool* with new functionality and with an up-to-date description of its processing, especially in the form of metadata files, and we have undertaken tasks to implement a distributed processing architecture based on XML SOAP messages passed between *geotool* and seismic processing routines running as Web services, possibly on different computers and different operating systems.

Software Requirements for a Reference Event Characterization Tool

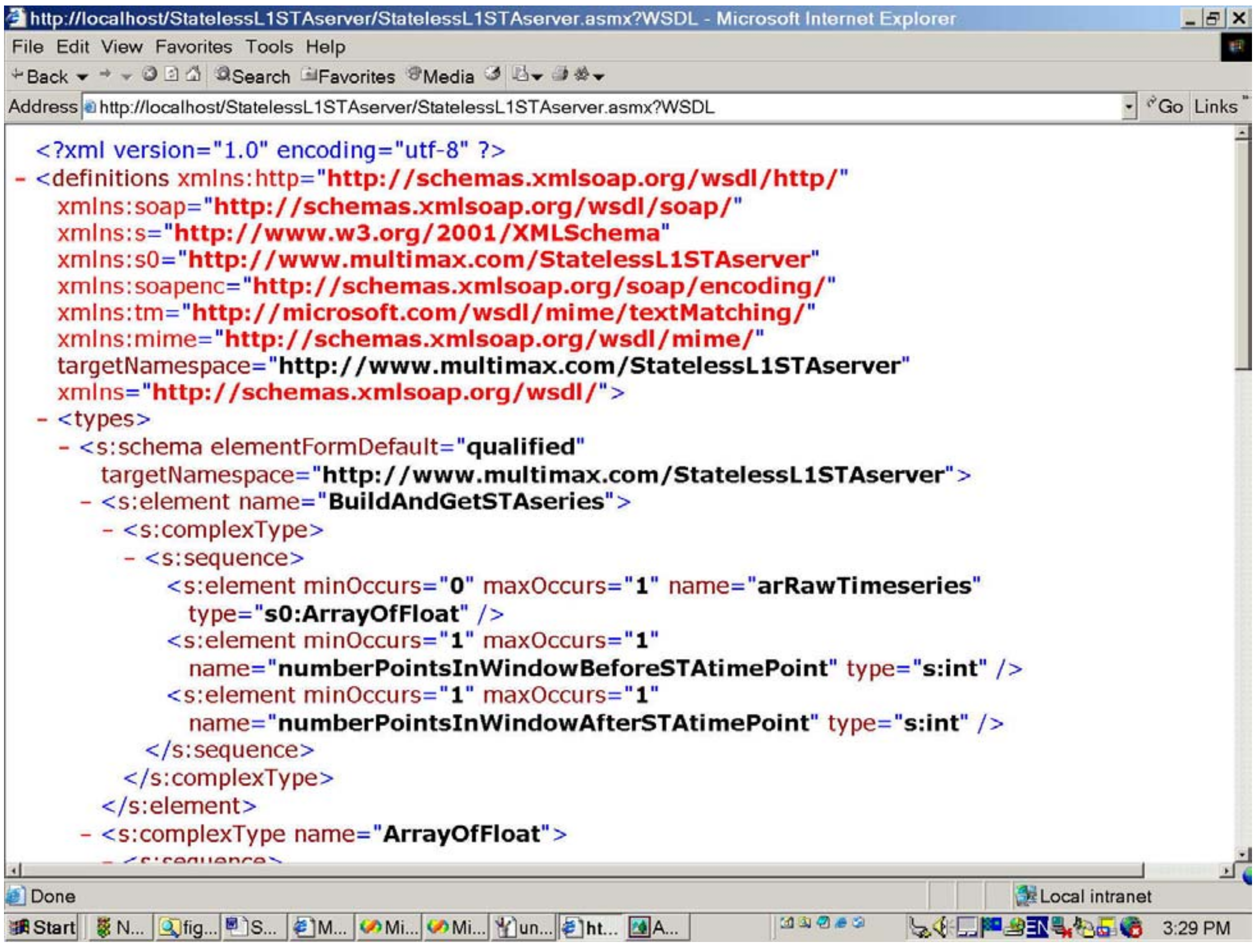
We have undertaken an extensive review of the operational capabilities, limitations, problems, and deficiencies of the stand-alone seismic analysis program *geotool*. The program itself (Henson, 1993) is now ten-years old, and although it has continued to evolve throughout its software life cycle, its most recent publicly distributed documentation (Coyne and Henson, 1995) is eight-years old. We have written a documentation update describing the program's current operations, and we have constructed a software requirements specification detailing the changes to the program that will be necessary if it is to be used as a central component of the reference event analysis system. We have also categorized the current operations of *geotool* in terms of a metadata description of the results of the processing, as will be described below.

Change in System Architecture

In our previous work (Rivers et al., 2002) we investigated the use of CORBA as the backbone architecture for data processing communications between *geotool* and remote programs. In our current work we have replaced CORBA with XML SOAP Web services as the backbone of the distributed processing. There are several benefits motivating this change, but there are also certain costs associated with it. Within a single computer, it may be preferable to implement inter-thread communications using direct socket connections for remote procedure calls (RPCs), and, within a client/server LAN, it may be preferable to use a binary protocol such as CORBA, as has been done throughout the 1990s to connect PCs to mainframes. For business-to-business commerce, however, it is preferable to invoke RPCs and transmit data using ASCII messages sent across the Internet through HTTP, a format that can work even if the client and server operate inside separate firewalls. It is for this reason that XML SOAP Web services are rapidly becoming the industry standard, and we feel that a reference event analysis system can make use of this loosely coupled system by swapping newly developed software into and out of it. Moving from sockets to CORBA to SOAP results in a gain of flexibility by moving to increasingly higher levels of software abstraction within the TCP/IP stack, but it comes at the cost of decreased efficiency, as direct binary communications are replaced by ASCII messages. We nevertheless believe this loss in efficiency is offset by the decrease in development time, and especially by the decrease in software maintenance time, required for the implementation of such a loosely coupled system of independently developed components rather than a large single program or a tightly coupled client/server system. Another advantage is that whereas CORBA permits communication among the UNIX, Linux, and Java platforms, SOAP Web services permit these platforms to communicate easily with the Windows 2000 platform.

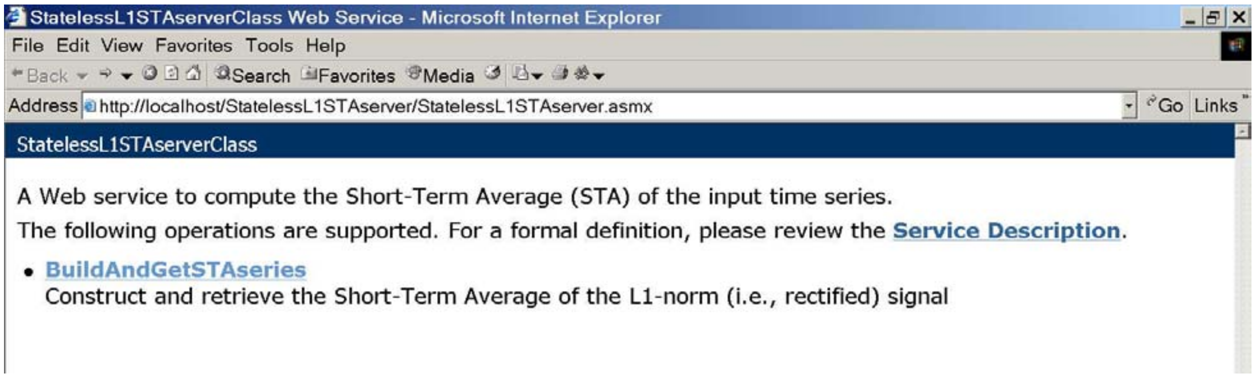
Our investigations of the use of XML SOAP for distributed seismic processing will be illustrated by a Web service that computes the short-term average (STA) of a time series. This routine is a C++ program running within the Microsoft .NET software platform under Windows 2000 (not as a program running on the Java platform within Windows). Figure 1 shows the Web Services Description Language (WSDL) interface that this program presents to the Internet. In this case the WSDL file was generated automatically by the Microsoft Visual Studio software development tool when the C++ program was installed within the *Inetpub/wwwroot* directory of the Web server, so the programmer does not need to write the WSDL description. Figure 2 shows the format of the SOAP message that should be used by a client program such as *geotool* that has need of this particular Web service. The Microsoft.NET development environment automatically generates this SOAP format tool, and SOAP messages in that format can then be used by a program running under Linux (like *geotool*) to communicate with this Windows program.

In our description of the work that we performed using CORBA as the middleware platform for distributed processing (Rivers et al., 2002), we presented a number of screenshots showing the transmission of waveforms between the C-language program *geotool* running under Linux and a Java program running within Windows. We shall not show a similar sequence of screenshots herein, since externally the distributed processing appears the same whether CORBA or Web services are used as the architecture. We do show in Figure 3 how to establish a routing to that same Java program, now accepting SOAP messages rather than CORBA transmissions to invoke its operations, and at the bottom of the figure we show that a *geotool* menu item is being used to receive the results of the processing performed by that remote application.


http://localhost/StatelessL1STAserver/StatelessL1STAserver.asmx?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/StatelessL1STAserver/StatelessL1STAserver.asmx?WSDL
Go Links
Done
Local intranet
Start
3:29 PM

```
  <?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://www.multimax.com/StatelessL1STAserver"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://www.multimax.com/StatelessL1STAserver"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://www.multimax.com/StatelessL1STAserver">
      - <s:element name="BuildAndGetSTAseries">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="arRawTimeseries"
                type="s0:ArrayOfFloat" />
              <s:element minOccurs="1" maxOccurs="1"
                name="numberPointsInWindowBeforeSTAtimePoint" type="s:int" />
              <s:element minOccurs="1" maxOccurs="1"
                name="numberPointsInWindowAfterSTAtimePoint" type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:complexType name="ArrayOfFloat">
```

Figure 1. Screenshot of a browser window displaying the WSDL (Web Services Description Language) interface to the Web service *StatelessL1STAserver*, which computes the short-term average (STA) of a time series, using the L1 (*i.e.*, absolute value) norm to rectify the data. *StatelessL1STAserver* is a C++ class operating as a Web service within the Microsoft .NET software platform under the Windows 2000 operating system. The WSDL interface shows the input values (namely the array of data points that make up the time series, along with a couple of integers describing the desired STA time window) that can be used by *geotool* (or any other program needing the service of this STA algorithm) to access this Web service, as well as the output values (namely, a new array of data points) that this service will return to the calling program. The application that needs this service will call a local proxy object that has the same interface as is shown in this WSDL file, and the proxy will transmit the request via the Web server (within the local host, or across the Intranet or Internet, as directed by the URL for the Web service) and will receive the response. Because WSDL and SOAP are XML vocabularies, it does not matter that *geotool* and *StatelessL1STAserver* are written in different programming languages and run on different operating systems.

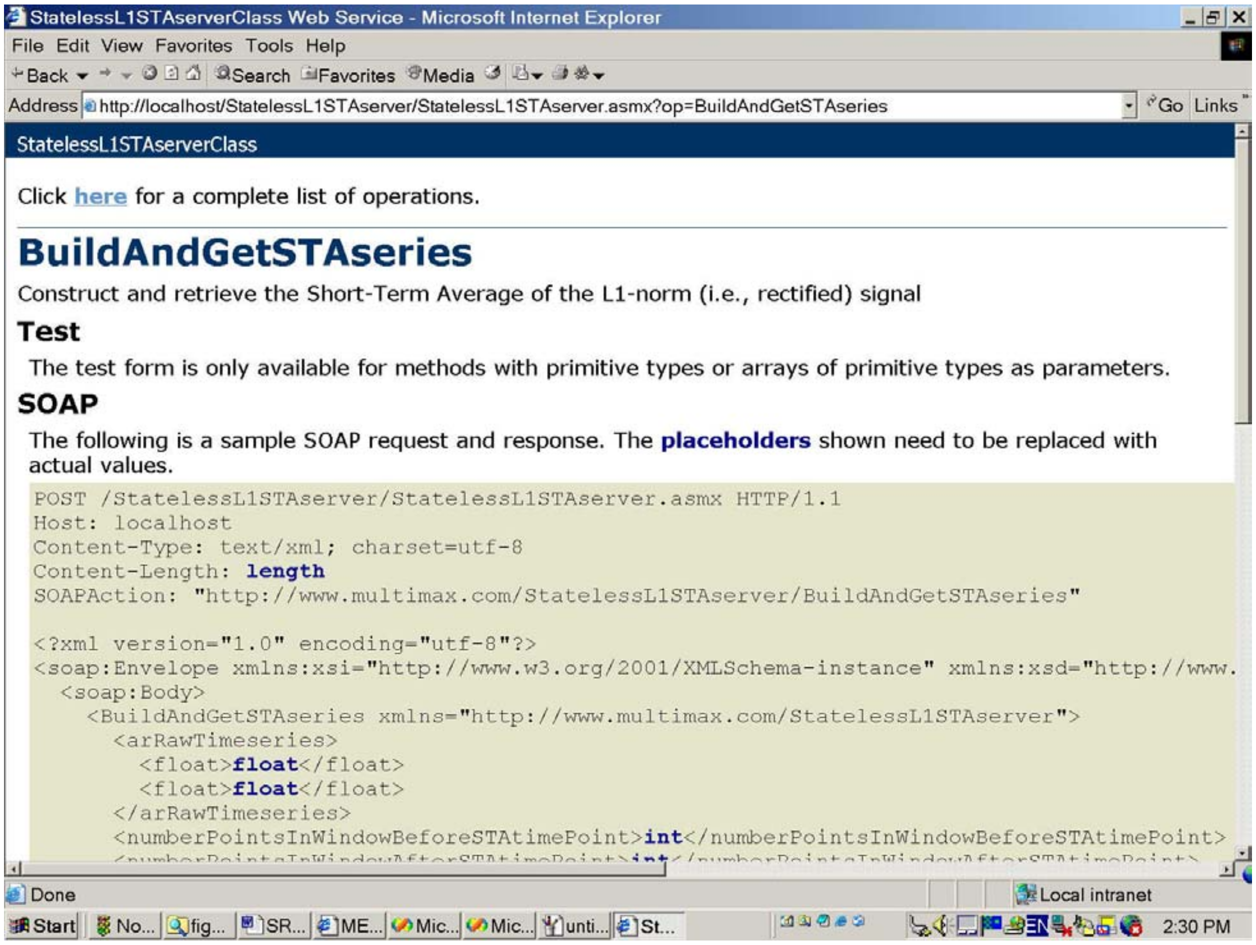

StatelessL1STAserverClass Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/StatelessL1STAserver/StatelessL1STAserver.asmx
Go Links

StatelessL1STAserverClass

A Web service to compute the Short-Term Average (STA) of the input time series.

The following operations are supported. For a formal definition, please review the Service Description.

- BuildAndGetSTAseries
  Construct and retrieve the Short-Term Average of the L1-norm (i.e., rectified) signal


StatelessL1STAserverClass Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/StatelessL1STAserver/StatelessL1STAserver.asmx?op=BuildAndGetSTAseries
Go Links
Done
Local intranet
Start No... fig... SR... ME... Mic... Mic... unti... St...
2:30 PM

StatelessL1STAserverClass

Click here for a complete list of operations.

BuildAndGetSTAseries

Construct and retrieve the Short-Term Average of the L1-norm (i.e., rectified) signal

Test

The test form is only available for methods with primitive types or arrays of primitive types as parameters.

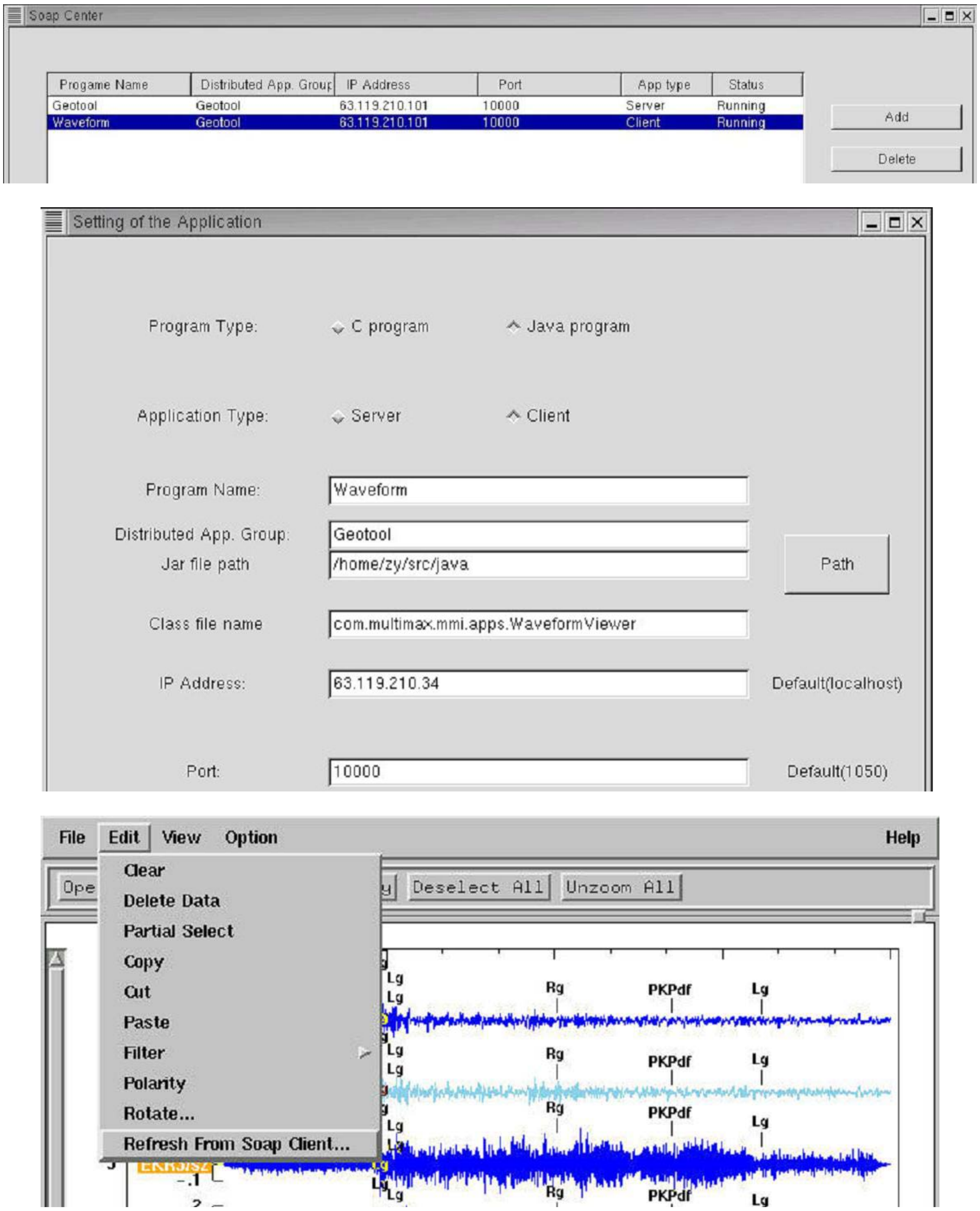
SOAP

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /StatelessL1STAserver/StatelessL1STAserver.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.multimax.com/StatelessL1STAserver/BuildAndGetSTAseries"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
  <soap:Body>
    <BuildAndGetSTAseries xmlns="http://www.multimax.com/StatelessL1STAserver">
      <arRawTimeseries>
        <float>float</float>
        <float>float</float>
      </arRawTimeseries>
      <numberPointsInWindowBeforeSTAtimePoint>int</numberPointsInWindowBeforeSTAtimePoint>
```

Figure 2. (Top) Screenshot showing a browser window that exposes the *StatelessL1STAserver* Web service to examination by a potential user of this service. The hyperlink to the "Service Description" exposes the WSDL file shown in Figure 1 that can be used to construct the interface to the proxy object. (Bottom) Screenshot showing that the hyperlink to the *BuildAndGetSTAseries* method of the C++ class exposes a template of the Simple Object Access Protocol (SOAP) message that will be sent to the Web service via HTTP when the calling program, such as *geotool*, makes a call to the *BuildAndSetSTAseries* method of the proxy object.



Soap Center

Progame Name	Distributed App. Group	IP Address	Port	App type	Status
Geotool	Geotool	63.119.210.101	10000	Server	Running
Waveform	Geotool	63.119.210.101	10000	Client	Running

Setting of the Application

Program Type: C program [✓ Java program]

Application Type: Server [✓ Client]

Program Name: Waveform

Distributed App. Group: Geotool

Jar file path: /home/zy/src/java

Class file name: com.multimax.mmi.apps.WaveformViewer

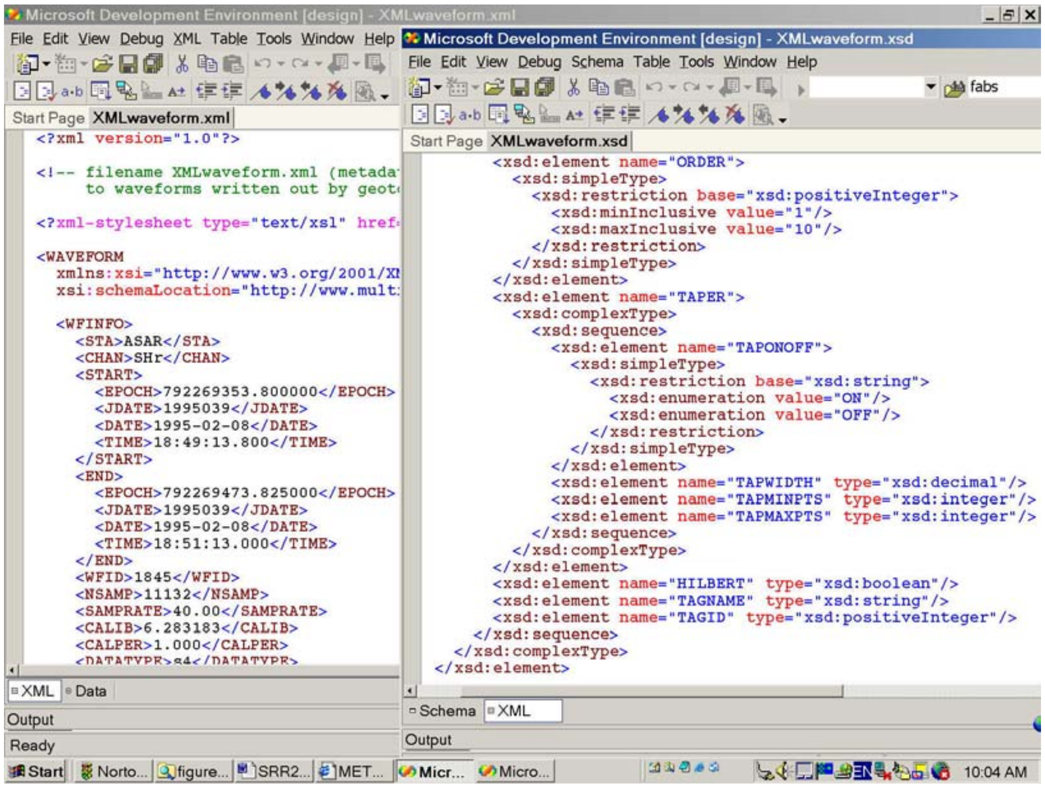
IP Address: 63.119.210.34 Default(localhost)

Port: 10000 Default(1050)

Figure 3. (Top and middle) Windows that set up a SOAP RPC connection from the C-language program *geotool* to the Java program *Waveform*. Although in this instance both programs are running on the same computer, this need not be the case. (Bottom) A window showing the menu item to refresh the selected waveform in the *geotool* display using data modified within the *Waveform* program.

Metadata Requirements

In seismic data analysis the issue of metadata is always an important one, since before the data can be entered into the Knowledge Base and the events can be located, parameterized, and classified it is necessary to know such things as how the raw data were obtained, how reliable were the measurements, what automatic and interactive processing were applied to the data, and the thresholds beyond which the processing results are invalid . This need for a thorough description of the metadata is even more nearly critical for a distributed processing system than for a single large seismic analysis package, since in our reference event system the client program such as *geotool* will treat the independent analysis programs as black boxes. It will send them data and receive results back, but the internal operations of these remote services must be regarded as unknown. It is therefore essential that the servers return to the clients not only the output of their algorithms but a thorough metadata description of all the values it returns and how they were derived. We have therefore undertaken the design of metadata records for all the processing routines within *geotool*, and we are extending that design to include external processing that will communicate with *geotool*. XML is becoming the software industry standard to use for categorizing metadata, and this format is especially beneficial to us since it can be passed back and forth between clients and Web services through HTTP, along with the XML SOAP RPC messages that instigate the data processing. Because ORACLE and other RDBMS vendors are enabling their products to manage XML metadata records along with relational tables, we shall be able to store on the server the contents of the passed metadata files. Figure 4 shows the XML markup for a metadata file describing the waveform processing applied by *geotool*. Because the XML markup tags are by definition extensible, we must define the tags that will be used in all instances of waveform processing, and the schema for these tags (itself an XML document) is shown on the right-hand side of the figure. The design of this particular schema is illustrated in Figure 5. Although the XML markup is intended to be read, principally by the data processing software, it can be presented in a form readily accessible by the *geotool* user, as is shown in Figure 6. The stylesheet that transforms the XML file into a presentation format is yet another XML document, and it is shown on the right-hand side of that figure.


Microsoft Development Environment [design] - XMLwaveform.xml
File Edit View Debug XML Table Tools Window Help
Start Page XMLwaveform.xml
XML Data
Output
Ready
Microsoft Development Environment [design] - XMLwaveform.xsd
File Edit View Debug Schema Table Tools Window Help
fabs
Start Page XMLwaveform.xsd
Schema XML
Output
Start Norto... figure... SRR2... MET... Micr... Micro... 10:04 AM

```
<?xml version="1.0"?>

<!-- filename XMLwaveform.xml (metada
     to waveforms written out by geot

<?xml-stylesheet type="text/xsl" href

<WAVEFORM
  xmlns:xsi="http://www.w3.org/2001/X
  xsi:schemaLocation="http://www.mult

  <WFINFO>
    <STA>ASAR</STA>
    <CHAN>SHr</CHAN>
    <START>
      <EPOCH>792269353.800000</EPOCH>
      <JDATE>1995039</JDATE>
      <DATE>1995-02-08</DATE>
      <TIME>18:49:13.800</TIME>
    </START>
    <END>
      <EPOCH>792269473.825000</EPOCH>
      <JDATE>1995039</JDATE>
      <DATE>1995-02-08</DATE>
      <TIME>18:51:13.000</TIME>
    </END>
    <WFID>1845</WFID>
    <NSAMP>11132</NSAMP>
    <SAMPRATE>40.00</SAMPRATE>
    <CALIB>6.283183</CALIB>
    <CALPER>1.000</CALPER>
    <DATATYPE>s4</DATATYPE>
```

```
          <xsd:element name="ORDER">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:minInclusive value="1"/>
                <xsd:maxInclusive value="10"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="TAPER">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="TAPONOFF">
                  <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                      <xsd:enumeration value="ON"/>
                      <xsd:enumeration value="OFF"/>
                    </xsd:restriction>
                  </xsd:simpleType>
                </xsd:element>
                <xsd:element name="TAPWIDTH" type="xsd:decimal"/>
                <xsd:element name="TAPMINPTS" type="xsd:integer"/>
                <xsd:element name="TAPMAXPTS" type="xsd:integer"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="HILBERT" type="xsd:boolean"/>
          <xsd:element name="TAGNAME" type="xsd:string"/>
          <xsd:element name="TAGID" type="xsd:positiveInteger"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
```

Figure 4. The XML markup describing a waveform (characterized by a CMR-format *.wfdisc* database record) that has undergone interactive processing by geotool is shown in the window on the left-hand side of the screenshot. The XML schema that was used for this metadata file is shown in the window on the right-hand side of the screenshot.

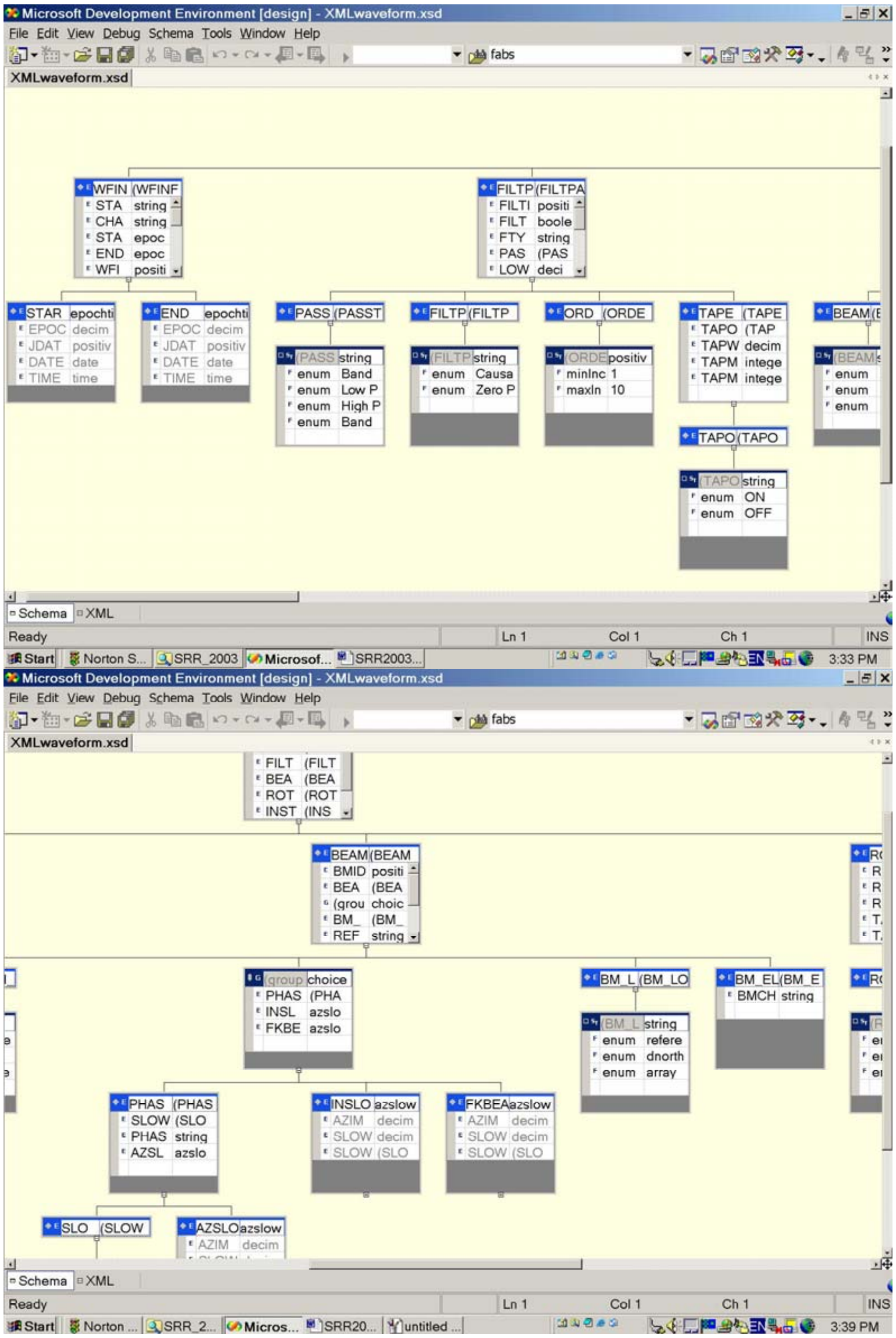
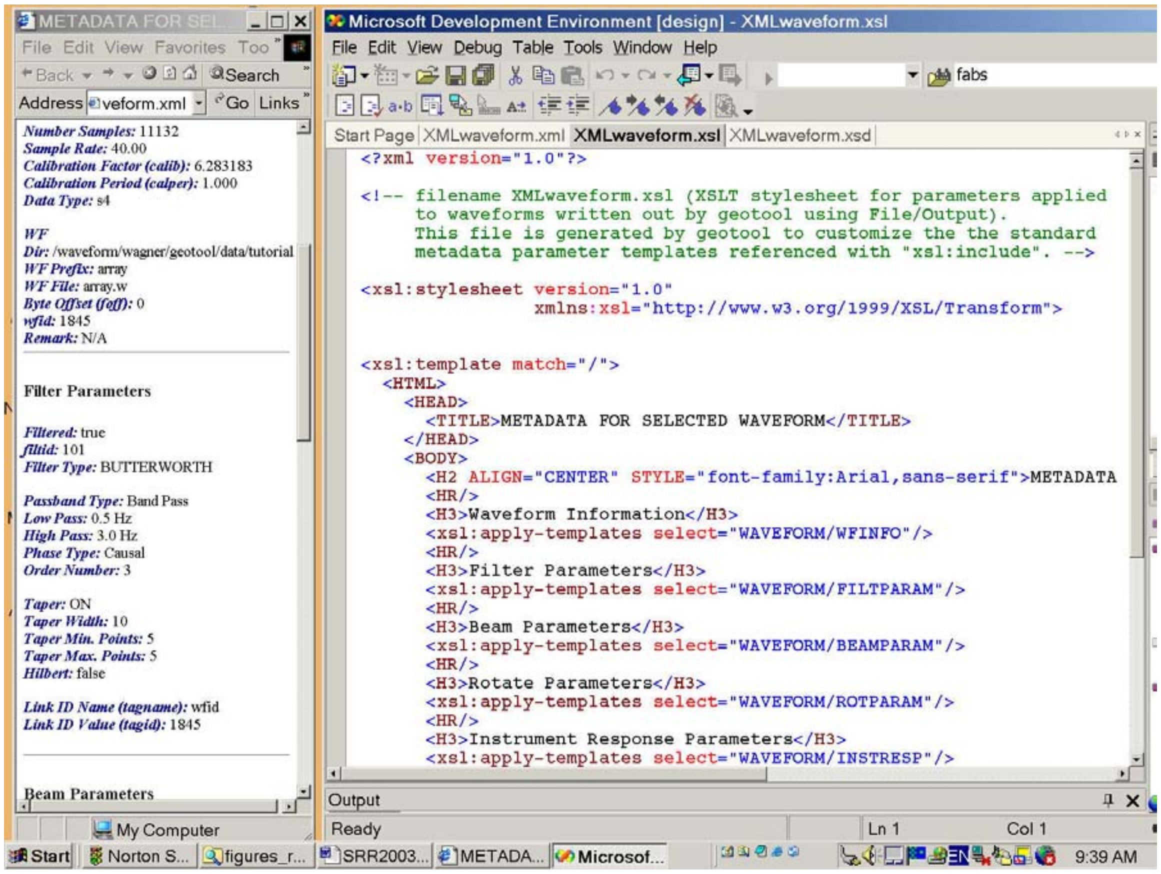


Figure 5. (Top and bottom) Two selected excerpts from the design layout of the waveform metadata XML schema that was shown in the window on the right-hand side of the screenshot in Figure 4.


METADATA FOR SEL...
File Edit View Favorites Too
Back Search
Address veform.xml Go Links
My Computer
Microsoft Development Environment [design] - XMLwaveform.xsl
File Edit View Debug Table Tools Window Help
fabs
Start Page XMLwaveform.xml XMLwaveform.xsl XMLwaveform.xsd
Output
Ready
Ln 1
Col 1
Start Norton S... figures_r... SRR2003... METADA... Microsof...
9:39 AM

Number Samples: 11132
Sample Rate: 40.00
Calibration Factor (calib): 6.283183
Calibration Period (calper): 1.000
Data Type: s4

WF
Dir: /waveform/wagner/geotool/data/tutorial
WF Prefix: array
WF File: array.w
Byte Offset (foff): 0
wfid: 1845
Remark: N/A

Filter Parameters

Filtered: true
filtid: 101
Filter Type: BUTTERWORTH

Passband Type: Band Pass
Low Pass: 0.5 Hz
High Pass: 3.0 Hz
Phase Type: Causal
Order Number: 3

Taper: ON
Taper Width: 10
Taper Min. Points: 5
Taper Max. Points: 5
Hilbert: false

Link ID Name (tagname): wfid
Link ID Value (tagid): 1845

Beam Parameters

```
<?xml version="1.0"?>

<!-- filename XMLwaveform.xsl (XSLT stylesheet for parameters applied
     to waveforms written out by geotool using File/Output).
     This file is generated by geotool to customize the the standard
     metadata parameter templates referenced with "xsl:include". -->

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


<xsl:template match="/">
  <HTML>
    <HEAD>
      <TITLE>METADATA FOR SELECTED WAVEFORM</TITLE>
    </HEAD>
    <BODY>
      <H2 ALIGN="CENTER" STYLE="font-family:Arial,sans-serif">METADATA
      <HR/>
      <H3>Waveform Information</H3>
      <xsl:apply-templates select="WAVEFORM/WFINFO"/>
      <HR/>
      <H3>Filter Parameters</H3>
      <xsl:apply-templates select="WAVEFORM/FILTPARAM"/>
      <HR/>
      <H3>Beam Parameters</H3>
      <xsl:apply-templates select="WAVEFORM/BEAMPARAM"/>
      <HR/>
      <H3>Rotate Parameters</H3>
      <xsl:apply-templates select="WAVEFORM/ROTPARAM"/>
      <HR/>
      <H3>Instrument Response Parameters</H3>
      <xsl:apply-templates select="WAVEFORM/INSTRESP"/>
```

Figure 6. The browser window on the left-hand side of the screenshot shows, in a format that can readily be understood by the *geotool* operator, the contents of the waveform metadata file that were shown in XML format in the window on the left-hand side of the screenshot in Figure 4. To the right of the browser window there is shown another window that displays the XSLT (Extensible Stylesheet Language – Transformation) file that was used to transform the XML metadata file into the visual presentation form that is shown in the browser window.

<u>CONCLUSIONS AND RECOMMENDATIONS</u>

Even if it were feasible to construct an expanded version of *geotool* that offered all the capabilities needed by a scientist to characterize fully the seismic reference events that will be included in the Knowledge Base, such a program would suffer from inflexibility to change in the face of new requirements or the development of new seismic data analysis techniques. We have therefore examined the use of a system architecture that is based on independent software components that may be located on a single computer or distributed across a network (possibly the Internet). We have found the design and construction of such a system architecture to be possible using the modern software technology which is employed in commercial applications such as business-to-business commerce. Although our initial investigations demonstrated that CORBA technology is suitable for that purpose, our work within the last year has shown that the technology of XML SOAP Web services can also serve as the foundation for a reference event analysis system. In such a system a graphically intensive application such as *geotool* can act as the user interface, dispatching data and commands to independent software components that may be written in different programming languages and that may run under different operating systems. It is our recommendation that XML Web services rather than CORBA be used as the foundation for the reference event analysis system, since this appears to be the prevalent trend in the software industry and since it is much easier to communicate with applications running under the Windows operating system from applications running on different platforms by means of XML and SOAP than it is by means of a software bridge between CORBA and Windows.

In a distributed data processing system, the generating, transmitting, and archiving of files of metadata describing the seismic analysis process is even more important than in a stand-alone application used for analysis. We have therefore conducted a detailed analysis of the metadata that describe the output of *geotool*, and we have examined how XML can be used as the format for describing those metadata. We recommend that XML metadata files, schemas, and stylesheets be designed for all software components that will be used within the reference event analysis system.

We envision that a reference event analysis system will consist of a graphical presentation tier running on a desktop computer or workstation for user interactions, a data tier that can run on a dedicated database server, and an applications tier that may run on the same computer as the user interface or that may be distributed across multiple computing platforms. Although initially *geotool* will be used for both the graphical presentation tier and the applications tier, the creation of new applications software as well as the incorporation of legacy seismic analysis programs into the applications tier will eventually reduce the need for the analysis capabilities of *geotool*. We anticipate that *geotool* will then be replaced by a user interface program written using an object-oriented design that will be easier to maintain throughout the remainder of the system's software lifecycle than is the design of a huge C-language program like *geotool*.

ACKNOWLEDGEMENTS

We are indebted to Floriana Ryall for her many extensive and insightful contributions to the evaluation of problems with the operation of *geotool* and the identification of useful and important extensions to its current capabilities.

REFERENCES

Coyne, J.M. and I. Henson (1995), *Geotool* Sourcebook: User's Manual, Phillips Laboratory report PL-TR-96-2021.

Henson, I. (1993), The *Geotool* Seismic Analysis System, in: Proceedings of the 15th Annual Seismic Research Symposium 8-10 September 1993, Phillips Laboratory report PL-TR-93-2160.

Rivers, D.W., C.A. Schultz, and D.A. Dodge (2002), Efficient Integration of Old and New Research Tools for Automating the Identification and Analysis of Seismic Reference Events, in: Proceedings of the 24th Seismic Research Review, Los Alamos National Laboratory report LA-UR-02-5048, 882-891 (available at http://www.nemre.nn.doe.gov/review2002/CD/screen/07-05.pdf).